Implementation of Delay Efficient ALU using Vedic Multiplier with AHL

P. Vimala, PhD
Associate Professor
Department of Electronics and Communication Engineering,
Dayananda Sagar College of Engineering,
Bangalore, India

Swapna M.S.
PG Scholar
Department of Electronics and Communication Engineering,
Dayananda Sagar College of Engineering,
Bangalore, India

ABSTRACT

Digital multipliers are most widely used component in applications such as convolution, Fourier transform, discrete cosine transforms, and digital filtering. Because outturn of these applications mainly depends on multiplier speed, therefore multipliers must be designed efficiently. In the proposed architecture, a variable-latency multiplier design with novel AHL architecture and a razor flip flop is used, which results in reduced delay and increased speed than the existing system. Meanwhile proposed architecture is used to compare array multiplier, column-bypassing multiplier, row-bypassing multiplier and Vedic multiplier. The experimental result shows that the Vedic multiplier has better performance in power consumption and delay. Here in this work Vedic multiplication is done using Urdhva Tiryakbhyam Sutra (Algorithm), which results in minimum delay. Thus using Vedic multiplier ALU is designed which results in enhanced performance compared to contemporary design.

General Terms

Adaptive hold logic (AHL), Urdhva-Tiryakbhyam sutra (Algorithm).

Keywords

Adaptive hold logic (AHL), negative bias temperature instability (NBTI), positive bias temperature instability (PBTI), variable latency, Vedic mathematics, Urdhva-Tiryakbhyam sutra (Algorithm).

1. INTRODUCTION

Digital Multiplication is most critical arithmetic functional unit. As multiplication predominates the operating time for most of the DSP applications like convolution, digital filtering, Fourier transform, discrete cosine transform etc. Since, their outputs are mainly based on the multipliers speed, and if multipliers are slow then the execution of the whole system will get diminished. So there is need of high speed multipliers.

As CMOS transistors are scaled down to ultra-deep submicron technologies, device accuracy can't be neglected. Negative Bias Temperature Instability (NBTI) effects are caused due to device aging process, which will have high impact on circuit performance and speed. The PMOS transistors threshold voltage increases due to NBTI effects over a time, due to this switching speed and circuit performance will be degraded. As a result of this circuit may fail because of timing variations. NBTI effect caused due to trap creation at Si/SiO2 interface at high temperature when the PMOS transistor is under negative bias i.e. Vgs = -Vdd and this will decreases the circuit driving current. The communication of hydrogen passivated Si atoms with inversion layer holes can break the Si – H bonds, thereby generating an interface trap.

These assembled interface traps between gate oxide interface and silicon results in the accessed threshold voltage (Vth), thereby decreasing the switching speed of the circuit. The reverse action takes place when the biased voltage is removed which will reduce the NBTI effect. But all generated interface traps cannot be eliminated by performing reverse operation. The same effect is caused to NMOS transistor when it is under positive bias which is called Positive Bias Temperature Instability (PBTI). The PBTI effect is much smaller when it is compared with NBTI effect and it is usually ignored. But for metal-gate NMOS transistors it is no longer ignored with remarkable charge confinement. Thus it's necessary to construct efficient high-speed multiplier.

Initially, [1] have proved that using NBTI-aware transistor sizing method can reduce the NBTI effect on the flip-flops timing features. Wu and Marculescu [2] implemented a restructuring function on basis of detecting logic symmetries and effects of transistors stacking. They also designed a method by considering path sensitization using NBTI method. In [3], [4], dynamic voltage scaling and body-basing techniques were proposed to reduce power or extend circuit life. These techniques, however, require circuit modification or do not provide optimization of specific circuits. Several variable-latency adders were proposed using the speculation technique with error detection and recovery [5]–[7].

The accuracy of hold logic is improved and to optimize the variable latency circuit [8] is done by proposing short path activation function. Later a variable-latency pipelined design [9], with a Booth algorithm was proposed. Ming-Chen Wen et al [10] have addressed a low power parallel multiplier architecture, where few columns of the multiplier arrays are switched-off when their outputs are known. There by saving switching power. The advantage of their work shows that it retains the original array architecture without initiating additional boundary cells as compared to earlier technique. J. Ohban, V. G. Moshnyaga [11] has addressed the digital multiplier based on dynamic bypassing of partial products. A Debasish Subudhi et al [12], [13], have proposed the design and implementation of high speed Vedic multiplier. They designed the multiplier using Urdhva Tiryakbhyam sutra. They designed the 4-bit modified multiplier using proposed 4-bit adder circuit. Their proposed multiplier gives delay of 12.825 ns which is less when it is compared with existing multiplier design. Ing-Chao Lin [14] has proposed the multiplier design with adaptive hold logic technique. His proposed multiplier uses the variable latency technique and he proved that using variable latency technique, the multiplier is capable to produce higher throughput and can adjust the AHL to reduce the performance degradation due to aging effect. He also proved that, his proposed architecture can applied to a column or row-bypassing multiplier.

The paper is organized as follows: section 2 introduces the background of array, column-bypassing, row-bypassing, Vedic multiplier and variable latency design. Description of the Proposed Methodology is given in Section 3 .The section 4 gives the detailed design of ALU using Proposed Multiplier Architecture. Experimental setup and results are presented in section 5. Section 6 concludes this paper.

2. VEDIC MATHEMATICS

The following The word 'Vedas' means store-house of all knowledge, from which the word 'vedic' is derived. Vedic mathematics is the most beneficial gift from the early sage's of India. This mathematics is easier and faster than that of conventional (modern) mathematics. In this mathematics although the calculations can be done in written from, it can also be done mentally. Therefore it is most interesting and creative way of performing mathematic calculations. After eight years of research on ancient Indian Veda's by Swamy Bharati Krishna Tirthaji (1884-1960), he comprised and reconstructed16 Vedas or algorithms and 16 upa-sutras or sub algorithms. Therefore Vedic mathematics calculation is mainly performed by using 16 basic sutras along with their upa-sutras (corollaries). Using these algorithms any numerical problems such as arithmetic, algebraic, geometric/ trigonometric functions can be resolved mentally.

Below gives the list of 16 sutras with their brief meanings in alphabetical order:

1. (Anurupye) Shunyamanyat –If one is in ratio, the other is zero.
2. Chalana Kalanabyham – Differences and Similarities.
3. Ekadhinkina Purvena – By one more than the previous one.
4. Ekanyunena Purvena – By one less than the previous one.
5. Gunakasamuchyah – The factor of the sum is equal to the sum of the factors.
6. Gunitasamuchyah – The product of the sum is equal to the sum of product.
7. Nikhilam Navatashcaramam Dashatah – All from 9 and the last from 10.
8. Paraavartya Yojayet – Transpose and adjust.
9. Puranapuranabyham – By the completion or non-completion.
10. Sankalana-vyavakalanabhyam – By addition and by subtraction.
11. Shesanyankena Charamena – The remainder by the last digit.
12. Shunyam Saamyasamuccaye – When the sum is the same, that sum is zero.
13. Sopaantyadvayamantyam – The ultimate and twice the penultimate.
14. Urdhva-Tiryagbyham – Vertically and crosswise.
15. Vyashtisamanstih – Part and whole.
16. Yaavadunam – Whatever the extent of its deficiency

2.1 Vedic Multiplications

Vedic multiplication is banked on the Vedic multiplication sutras/algorithms. Multiplications of two decimal/binary numbers are performed using these algorithms/sutras. This multiplication technique will result in computational power saving. Here Vedic multiplier is designed using Urdhva-Tiryagbyham sutra to perform the parallel multiplication.

2.2 Urdhva-Tiryagbyham sutra

It is a conventional multiplication algorithm applicative to all types of multiplication/division of one large by large numbers. Urdhva-Tiryagbyham means vertical and crosswise multiplication. This multiplication is depends on the novel idea, that will first generates all the partial products and then it will add these partial products concurrently. Figure 1, depicts the multiplication using Urdhva-Tiryagbyham sutra method. Figure 2, illustrates the example of multiplication of two decimal numbers 3451 and 6723 using vertical and crosswise method. The main advantage of this multiplication technique when compared with other multiplier is, has the number of bits increases, area and gate delay increases moderately. This multiplier is independent of clock frequency since parallel computation of partial products and their sums are done. Thus it is power, space and time efficient multiplier technique.

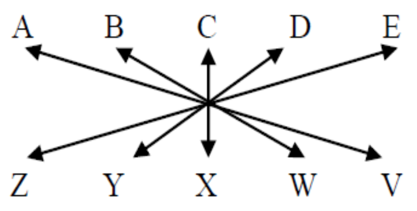
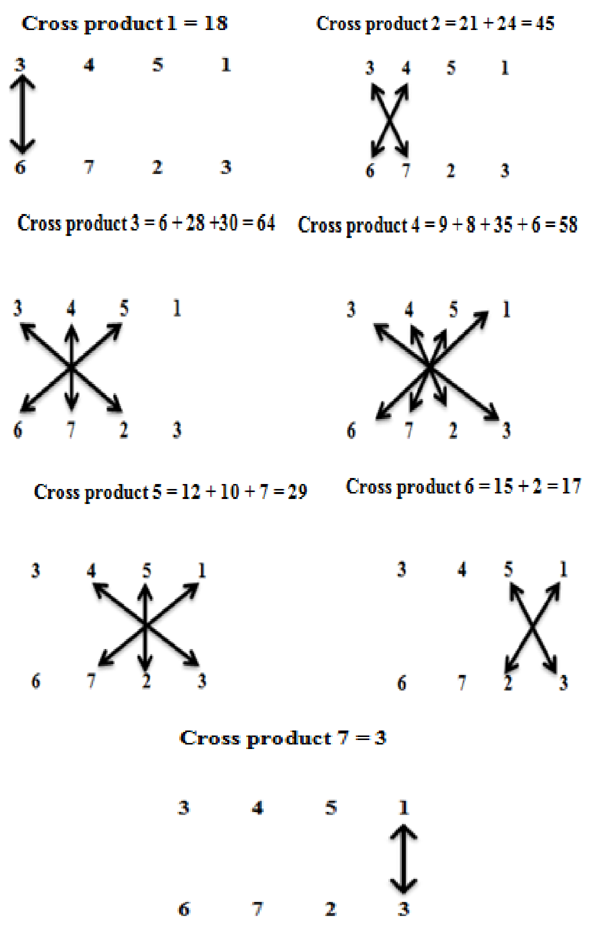
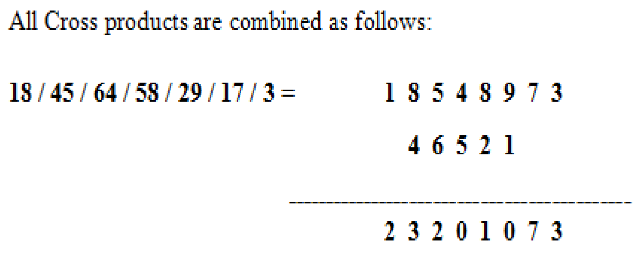


Fig 1: 4X4 Urdhava-Tiryabyham Sutra method

Cross-products are obtained as follows:


Cross product 1 = 18
Cross product 2 = 21 + 24 = 45
Cross product 3 = 6 + 28 +30 = 64
Cross product 4 = 9 + 8 + 35 + 6 = 58
Cross product 5 = 12 + 10 + 7 = 29
Cross product 6 = 15 + 2 = 17
Cross product 7 = 3
3 4 5 1
6 7 2 3



All Cross products are combined as follows:

```
18 / 45 / 64 / 58 / 29 / 17 / 3 =      1 8 5 4 8 9 7 3
                                          4 6 5 2 1
                                 -----------------------
                                        2 3 2 0 1 0 7 3
```

Fig 2: Example of multiplication of two decimal numbers 3451 and 6723 using vertical and crosswise method

2.3 Variable-Latency Design

Author Contemporary multipliers architecture uses the critical path cycles as an execution cycle period. Therefore significant timing waste will occur due to this critical path delay. Hence variable latency method is used in proposed multiplier design to minimize the timing waste present in the conventional multiplier circuit. Depending upon the number of cycles required by the multiplier, the variable latency method will divide the circuit into two paths namely: (1) Shorter path and (2) Longer path. Shorter path can execute exactly using one cycle, whereas longer path requires two cycles to execute. Thus when shorter paths are frequently activated, the latency's average of the variable-latency technique is more appropriate than that of conventional multiplier design which makes use of critical path as an execution period. In bypassing multipliers such as Column and Row-Bypassing technique, the path of the system is completely depends on the number of zero's present in the multiplicand and multiplicator bits. In the Column-Bypassing multiplier, delay distribution is left shifted as the number of zeros in the multiplicand bit increases there by reducing the average delay. This is because select line of the 2:1 multiplexer uses multiplicand bit to determine whether the input patterns needs one/two cycles to complete an operation. If more numbers of zeros are inserted in the multiplicand bit, which will reduce the number of switching activities of full adders used in the multiplier. Thus more number of full adders will be neglected and sum bit from the upper full adder is passed to lower full adder thus minimizing the delay in path. Similarly in case of Row-Bypassing multiplier, select lines of the 2:1 multiplexers use the multiplicator bit to decide one or two cycles are required by the input patterns to complete the operation. Therefore this will makes the bypassing multipliers as superior candidates for the design of variable latency technique.

3. DESCRIPTION OF THE PROPOSED METHODOLOGY

3.1 Low Power Vedic Multiplier with Adaptive Hold- logic

The basic block diagram of low-power Vedic multiplier with Adaptive Hold Logic (AHL) is shown in Figure 3. It consists of one Vedic multiplier, 1-bit Razor flip-flop and an Adaptive Hold Logic (AHL) circuit.

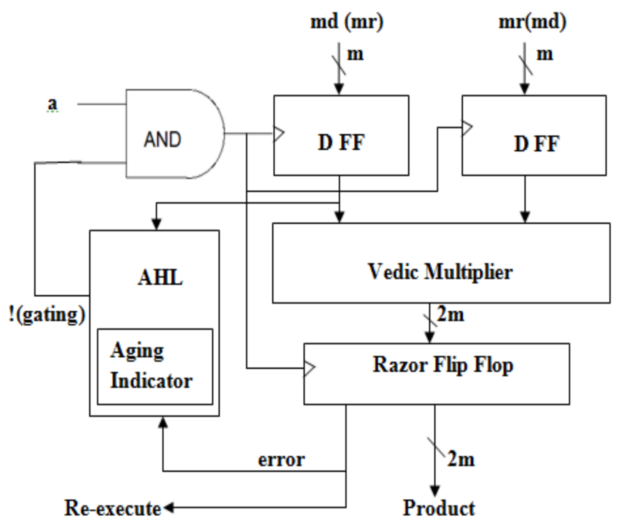


Fig 3: Low-Power Vedic Multiplier with AHL

3.2 Working of the propose multiplier

When the input pattern appears, the AHL circuit and the Vedic multiplier simultaneously execute their operations. The AHL circuit determines the number of clock cycles needed for the present input pattern according to the number of zero's in the multiplicand or multiplicator bits of the Vedic multiplier. If input pattern needs one cycle to complete, AHL outputs '1' for normal operation and AHL outputs'0' to disable the clock signal of the input flip-flops, if the input pattern needs two clock cycles to complete the operation. When Vedic multiplier completes its operation, the output of the multiplier block is fed to the Razor FF block. The Razor flip-flops (FF) block will examine the presence of timing violations for path delay. If timing variations present, which inform the system that the cycle period is not sufficient to complete the current operation and operation result of the multiplier block is inaccurate. Thus, it will inform the system that the current operation requires two cycles to run correctly, and thus Razor flip-flops will result an error signal to guarantee the operation is correct.

3.3 Adaptive Hold Logic (AHL)

The Adaptive Hold Logic (AHL) is the most essential part of the variable-latency multiplier design. Figure 4 depicts the block diagram of Adaptive Hold Logic.

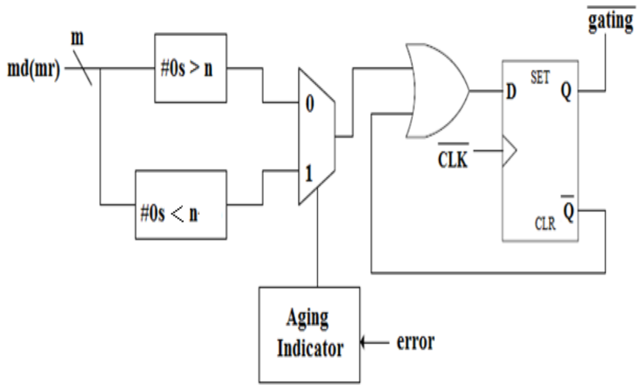


Fig 4: Adaptive Hold Logic (AHL)

The AHL circuit consists of two judging blocks, aging indicator, one 2:1 multiplexer and a D flip-flop. The main use of the aging indicator in AHL block is to display, whether the design has affected by remarkable performance reduction because of aging consequence. The simple counter is used to construct aging indicator, which computes the number of errors throughout a certain amount of operation and at the end it is reset to zero. The Vedic multiplier is inefficient to execute its operation if cycle period is too short, thereby causing timing violations. The Razor Flip-Flops will generate the error signals by capturing these timing violations. If errors occur regularly and rise above the predefined threshold voltage, which means that the design has affected by remarkable performance reduction because of aging consequence and therefore the aging indicator block will output signal 1, or else aging indicator block will output signal 0 to notify that aging consequence is still not important, and therefore no operations are required. Out of two judging blocks in the AHL architecture, first judging block will result signal 1 if the number of zero's in the multiplicand bits or multiplicator bits of the Vedic multiplier is considerably larger than 'n' (where 'n'- positive number), and second judging block will result signal 1 if the number of zero's in the multiplicand or multiplicator bits of the Vedic multiplier is lesser than 'n'. These two judging blocks are selected to determine if an input pattern needs one/two cycles to complete the operations, but out of two judging blocks 2:1 multiplexer will select only one at a time. In the beginning first judging block is used because, the aging consequence is not remarkable, and aging indicator will output signal zero. Over an extent of time when aging consequence becomes remarkable and produces output signal 1, the second judging block is selected. In contrast to first judging block, the second judging block enables a smaller number of patterns to get one-cycle pattern, because it requires more number of zero's in the multiplicand or multiplicator bits.

4. DESIGN OF ALU USING PROPOSED MULTIPLIER ARCHITECTURE

ALU stands for Arithmetic and Logic Unit; it is a digital circuit and performs all type of mathematical and logical functions hence it is referred as a fundamental building block of Central Processing Unit-CPU. Current generation CPU's are frequently operated at higher frequencies with minimized transistor size. Arithmetic and Logic Unit - ALU is one of the most critical and efficient block in CPU. Hence it is requirement to have fast and efficient ALU.

In this work, ALU is designed using two modules namely, multiplier module and addition and subtraction module as shown in the Figure 5. The16-bit Vedic multiplier is used which performs the parallel multiplication and conventional adder and subtractor module is used to perform addition and subtraction which is also 16-bit. In the above figure signals A and B are two 16-bit inputs and which is given as input to the Proposed Vedic Multiplier as well as Adder & subtractor module. Product is the output signal from multiplier module which is 32-bit. Similarly Sum and Difference are the two output signals from Adder & Subtractor module which is 17-bit.

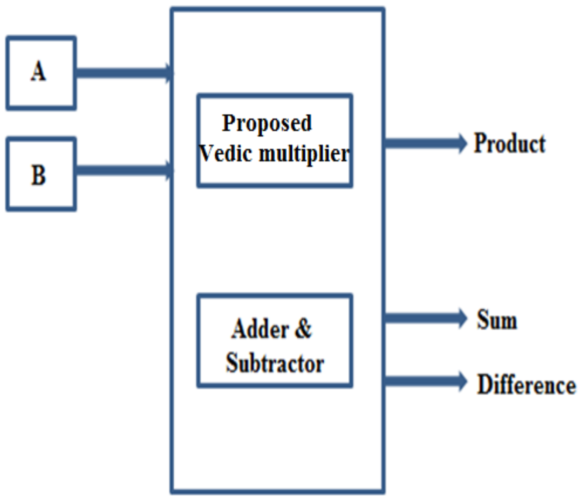


Fig 5: ALU Design using Proposed Vedic Multiplier

5. EXPERIMENTAL RESULTS

The experiments are conducted in a Xilinx ISE design suite 13.4 where Verilog code can be used for design implementation. Figure 6, shows the simulation result. System-level testing may be performed with the ISIM logic simulator, and such test programs must also be written in HDL languages. Test bench programs may include simulated input signal waveforms, or monitors which observe and verify the outputs of the device under test.

Figure 7, shows the delay comparison chart of existing and proposed multipliers. Table 1, shows the Comparison between 16x16 existing multipliers architecture and 16x16 proposed multipliers architecture.

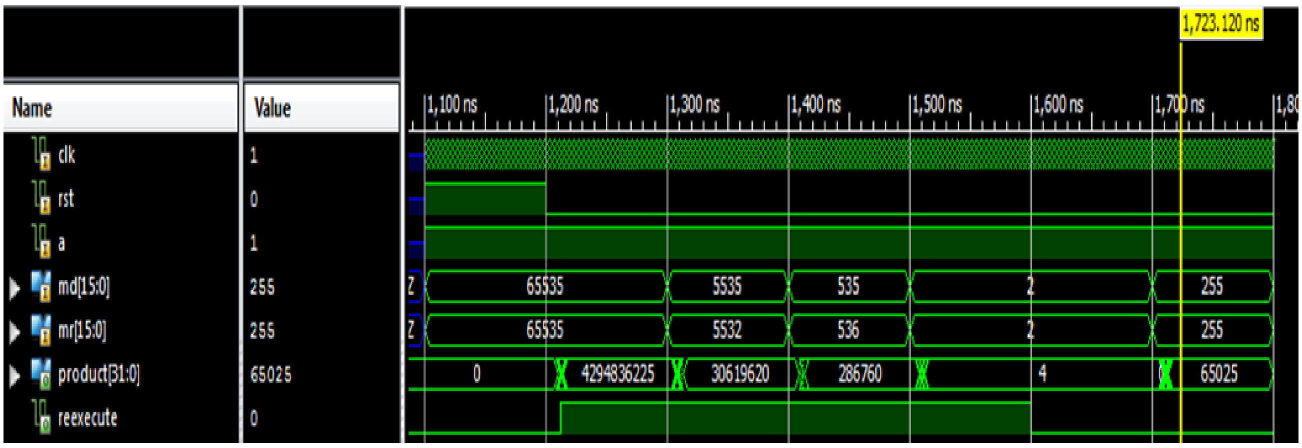


Fig 6: Simulation waveform of 16X16 Vedic Multiplier with AHL

Figure 8, shows the Delay and Logic Levels comparison chart of existing and proposed ALU. Table 2 gives the Comparison between existing and proposed ALU using 16x16 Vedic Multiplier architecture. Figure 9 Shows the FPGA Implementation of ALU using 16x16 Vedic Multiplier with AHL, which is implemented on Virtex 5 using Chipscope.

The table and graph shows that, the total logic and route delays are efficient for Proposed Vedic multiplier than other multipliers. Hence, the hardware implementation of ALU is performed using Proposed Vedic multiplier.

Table 1. Comparison between 16x16 existing multipliers architecture and 16x16 proposed multipliers architecture

Parameters	**Existing Array Multiplier**	**Proposed Array Multiplier**	**Existing Row-Bypassing Multiplier**	**Proposed Row-Bypassing Multiplier**	**Existing Column-Bypassing Multiplier**	**Proposed Column-Bypassing Multiplier**	**Existing Vedic Multiplier**	**Proposed Vedic Multiplier**
Total Delay (ns)	26.878	14.126	19.263	11.297	18.925	9.979	12.852	6.441
Logic Delay (ns)	5.246	2.546	4.730	1.858	4.644	1.772	2.667	2.025
Route Delay (ns)	21.632	11.580	14.533	9.439	14.281	8.207	10.185	4.415
Logic Levels	30	25	24	17	23	16	27	22
Total Memory Usage	311972 Kilobytes	345380 kilobytes	307940 kilobytes	341348 kilobytes	305316 kilobytes	341348 kilobytes	299684 kilobytes	338276 kilobytes

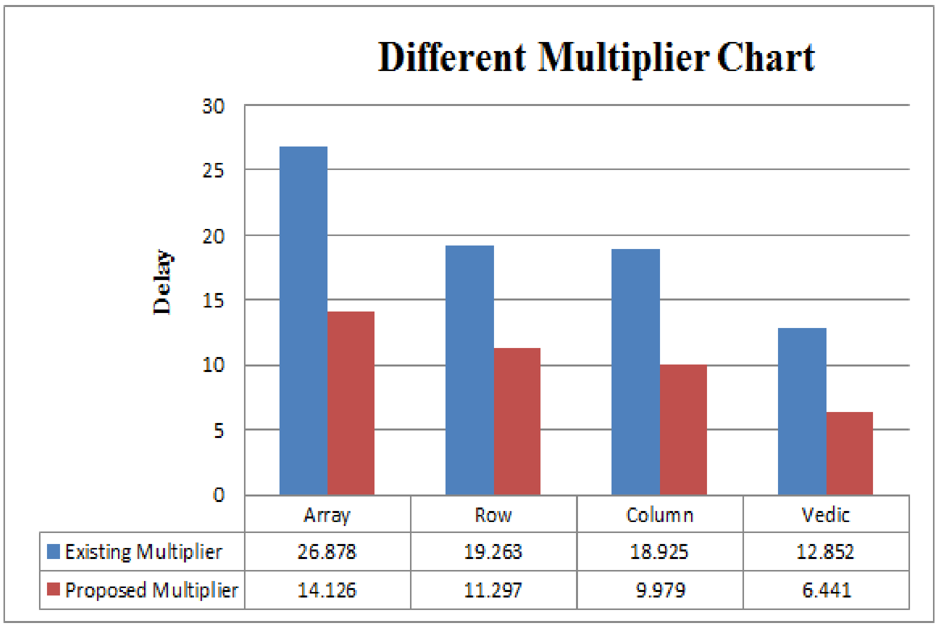
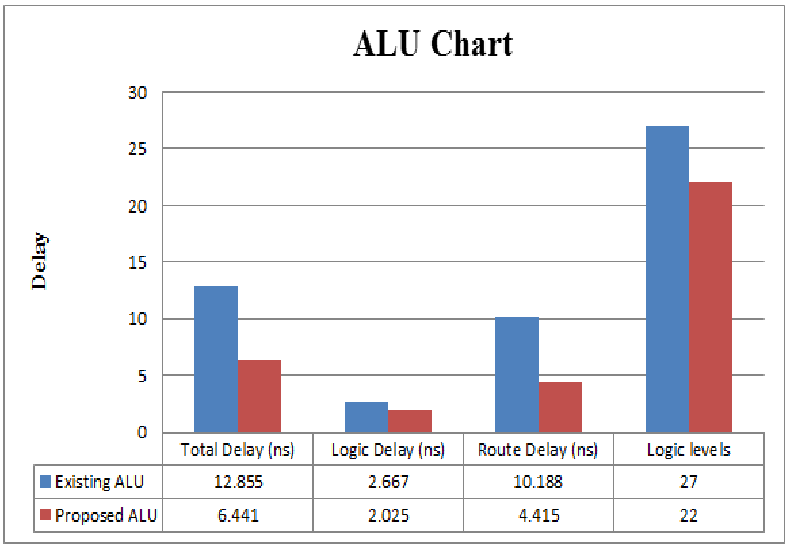


Fig 7: Delay comparison chart of various 16x16 Multipliers

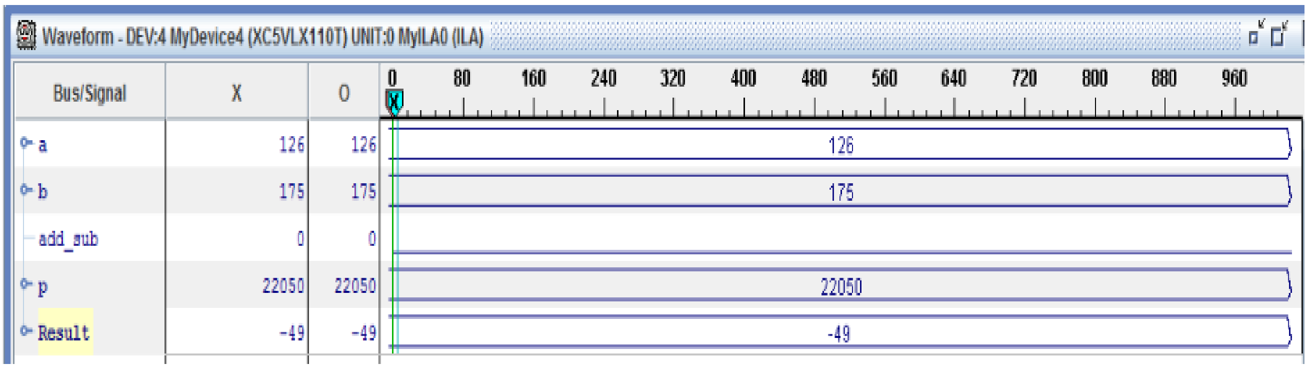
Table 2.Comparison between existing and proposed ALU using 16x16 Vedic Multiplier architecture

Parameters	**16x16 existing Vedic Multiplier**	**16x16 Proposed Vedic Multiplier**
Total Delay (ns)	12.855	6.441
Logic Delay (ns)	2.667	2.025
Route Delay (ns)	10.188	4.415
Logic Levels	27	22
Total Memory Usage	303572 kilobytes	342228 kilobytes
Power Utilization	1.210W	1.224W



	Total Delay (ns)	Logic Delay (ns)	Route Delay (ns)	Logic levels
🟦 Existing ALU	12.855	2.667	10.188	27
🟥 Proposed ALU	6.441	2.025	4.415	22

Fig 8: Delay and Logic Levels comparison chart of existing and proposed ALU



Bus/Signal	X	0	
a	126	126	126
b	175	175	175
add_sub	0	0	
p	22050	22050	22050
Result	-49	-49	-49

Fig 9: FPGA Implementation of ALU using 16x16 Vedic Multiplier with AHL

6. CONCLUSION

The design of 16-bit Array Multiplier, Row and Column-Bypassing multiplier, Vedic Multiplier using Urdhva Tiryakbhyam Sutra (Algorithm) and Arithmetic and Logic Unit has been realized on Virtex5 XC5VLX110T-2-FF1136 device. The computation delay for the existing ALU module is 12.855ns and Proposed ALU module with AHL technique is 6.441ns respectively, which clearly shows improvement in performance and speed. FPGA implementation using Virtex5 XC5VLX110T-2-FF1136 device demonstrates the hardware realization of ALU using proposed Vedic Multiplier with AHL technique. The proposed multiplier with Variable-Latency technique is able to adjust AHL to minimize the performance degradation. To reduce the delay and to increase the circuit speed, an extra modules such as Razor flip-flops and AHL module are added, which results in increase in area.

In this work Vedic Multiplier is designed using Urdhva Tiryakbhyam Sutra (Algorithm) this sutra results in generation of large number of partial products. To solve this problem multiplier can be designed using Toom Cook algorithm, which results in generation of fewer partial products.

In this work ALU is designed to perform multiplication, addition and subtraction. So future work can include logic functions and Multiply Accumulate Unit-MAC there by increasing the speed of the ALU.

7. REFERENCES

[1] H. Abrishami, S. Hatami, B. Amelifard, & M. pedram, (2008) “NBTI-Aware Flip-Flop Characterization and Design”, in Proc. 44th ACMGLSVLSI, pp.29-34.

[2] K. C. Wu & D. Marculescu, (2011) “Aging-Aware Timing Analysis and Optimization Considering path Sensitization”, in Proc. DATE, pp.1-6.

[3] Y. Lee & T. Kim, (2011) “A Fine-Grained Technique of NBTI-Aware Voltage Scaling and Body Biasing for Standard Cell Based Designs”, in Proc ASPDAC, pp.603-608.

[4] M. Basoglu, M. Orshansky, & M. Erez, (2010) “NBTI-Aware DVFS: A New Approach to Saving Energy and Increasing Processor Lifetime”, in Proc. ACMIEEEISLPED, pp.253-258.

[5] K. Du, P. Varman, & K. Mohanram, (2012) “High performance Reliable Variable Latency Carry Select Addition”, in Proc. DATE, pp. 1257-126.

[6] A.K. Verma, P. Brisk, & P. Ienne, (2008) “Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design”, in proc DATE, pp. 1250-1255.

[7] D. Baneres, J. Cortadella, & M. Kishinevsky (2009) “Variable-Latency Design by Function Speculation”, in Proc. DATE, pp. 1704-1709.

[8] Y. S. Su, D. c. Wang, S. C. Chang, & M. Marek-Sadowska (2011) "performance Optimization using Variable-latency Design Style", IEEE Trans. Very Large Scale Integr. (VLSI) Syst, vol. 19, pp.1874-1883.

[9] M. Olivieri, (2001) "Design of Synchronous and Asynchronous Variable-Latency Pipelined Multipliers", IEEE Trans. Very Large Scale Integr. (VLSI) Syst, vol.9, pp.365-376.

[10] M. C. Wen, S. J. Wang & Y. N. Lin, (2005) "Low Power Parallel Multiplier with Column Bypassing", in Proc. IEEE ISCAS,pp.1638-1641.

[11] J. Ohban, V. G. Moshnyaga, & K. Inoue, (2002) "Multiplier Energy Reduction through Bypassing of Partial products", in proc. APCCAS,pp. 13-17.

[12] A Debasish Subudhi, Kanhu Charan Gauda, Abinash kumar Pala & Jagmohan Das, (2012) "Design and Implementation of High Speed 4x4 Vedic Multiplier", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, No. 11,pp.362-366.

[13] Anvesh kumar et al, "Low Power ALU Design by Ancient Mathematics" (2010), The 2nd International Conference on Computer and Automation Engineering (ICCAE), Vol.5, pp. 862-865.

[14] Ing-Chao Lin, Member, IEEE, Yu-Hung Cho, & Yi-ming Yang,(2014) "Aging-Aware Reliable Multiplier Design with Adaptive Hold Logic", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol.23, No. 3 pp.1063-8210.